COMP5046
Natural Language Processing

Lecture 2: Word Embeddings and Representation

Semester 1, 2019

School of Computer Science
The University of Sydney, Australia

THE UNIVERSITY OF SYDNEY

Caren Han
Caren.Han@sydney.edu.au

0 LECTURE PLAN

THE UNIVERSITY OF SYDNEY

Lecture 2: Word Embeddings and Representation

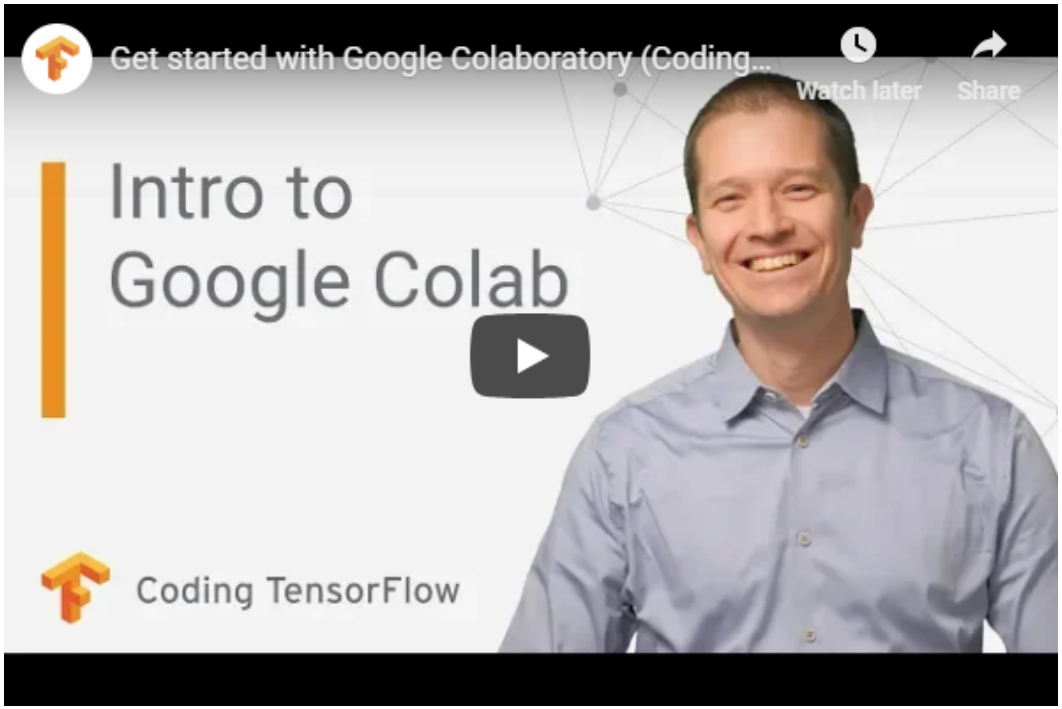
1. Lab Info
2. Word Meaning and Representation
3. Count based Word representation
    1. One-hot encoding
    2. Bag of Words
    3. Term Frequency-Inverse Document Frequency
4. Prediction based Word representation
    1. Word2Vec
    2. FastText
    3. Glove Word Vector
5. Next Week Preview

Info: Lab Exercise

What do we do during Labs?

In Labs, Students will use Google Co Lab

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory you can write and execute code, save and share your analyses, and access powerful computing resources, all for free from your browser.



https://colab.research.google.com/notebooks/welcome.ipynb

1 Info: Lab Exercise

Submissions

How to Submit

Students should submit **"ipynb" file** (Download it from "File" > "Download .ipynb") to Canvas.

When to Submit

Students must submit the Lab 1 in Week2 by the day before 5pm in Week3.

- If you were in **Tuesday 4pm or 5pm** Lab, you should submit your lab1 work by **Monday 5pm in week3**.
- If you were in **Wednesday 4pm or 5pm Lab**, you should submit your lab1 work by **Tuesday 5pm in week3**.
- If you were in **Thursday 5pm Lab**, you should submit your lab1 work by **Wednesday 5pm in week3**.

WORD REPRESENTATION

How to represent the meaning of the word?

Definition: meaning (Collins dictionary).

- the idea that it represents and which can be explained using other words.
- the thoughts or ideas that are intended to be expressed by it.

Commonest linguistic way of thinking of meaning:

signifier (symbol) ⟺ signified (idea or thing) = denotation

“Computer”

“Apple”

WORD REPRESENTATION

How do we have usable meaning in a computer?

- Common solution: Use e.g. WordNet, a thesaurus containing lists of synonym sets and hypernyms ("is a" relationships).
- http://wordnetweb.princeton.edu/perl/webwn

e.g. synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)',
'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
print("{}: {}".format(poses[synset.pos()],
", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g. hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

WORD REPRESENTATION

Problems with resources like WordNet

- Great as a resource but missing nuance
  - e.g. “proficient” is listed as a synonym for “good”. This is only correct in some contexts.
  - e.g. “glad” can be synonym for “fXXX off”???
- Missing new meanings of words
  - Impossible to keep up-to-date
- Subjective
- Requires human labor to create and adapt
- Difficult to compute accurate word similarity

0 LECTURE PLAN

THE UNIVERSITY OF SYDNEY

Lecture 2: Word Embeddings and Representation

1. Lab Info
2. Word Meaning and Representation
3. **Count based Word representation**
    1. One-hot encoding
    2. Bag of Words
    3. Term Frequency-Inverse Document Frequency
4. Prediction based Word representation
    1. Word2Vec
    2. FastText
    3. Glove Word Vector
5. Next Week Preview

COUNT based WORD REPRESENTATION

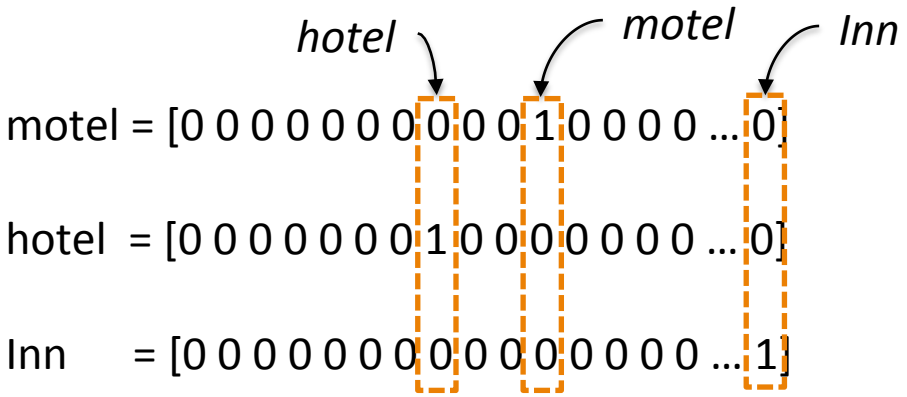
One-hot Encoding

- In traditional NLP, we regard words as discrete symbols.

Means one 1, the rest 0s

Words can be represented by **one-hot vectors**:

1. The categorical values be mapped to integer values (index)
2. Each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.



Vector dimension = number of words in vocabulary

3 COUNT based WORD REPRESENTATION

THE UNIVERSITY OF SYDNEY

Manual One-hot Encoding

Sentence Tokenization

```
import nltk
nltk.download('punkt')
word_data = "I enjoy studying natural language processing"
token = nltk.word_tokenize(word_data)
print (token)
```

Token Indexing

```
word2index={}
for voca in token:
    if voca not in word2index.keys():
      word2index[voca]=len(word2index)
print(word2index)
```

One-hot Encoding

```
def one_hot_encoding(word, word2index):
      one_hot_vector = [0]*(len(word2index))
      index=word2index[word]
      one_hot_vector[index]=1
      return one_hot_vector
```

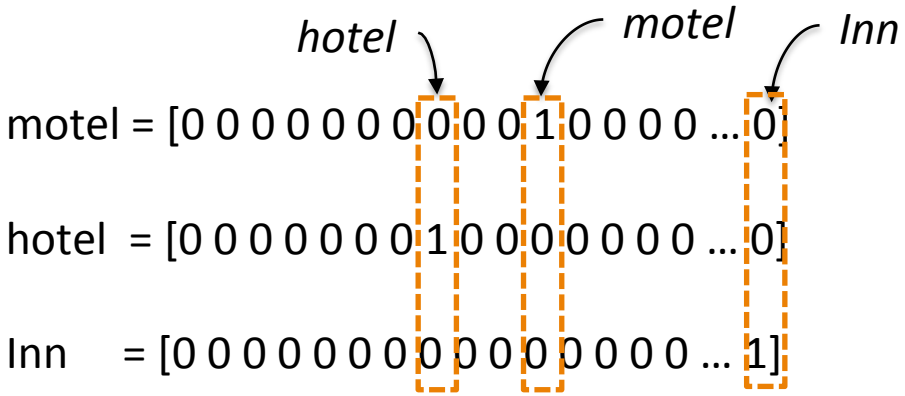
```
one_hot_encoding("enjoy",word2index)
```

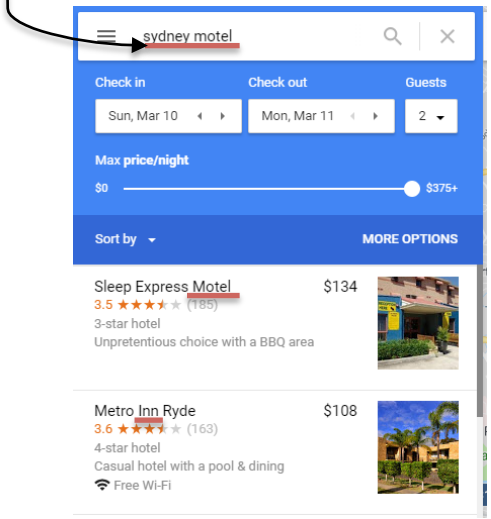
COUNT based WORD REPRESENTATION

Problem with one-hot vectors

Problem #1. No word similarity representation

Example: in web search, if user searches for “Sydney motel”, we would like to match documents containing “Sydney Inn”





There is no natural notion of similarity for one-hot vectors!

Problem #2. Inefficiency

Vector dimension = number of words in vocabulary

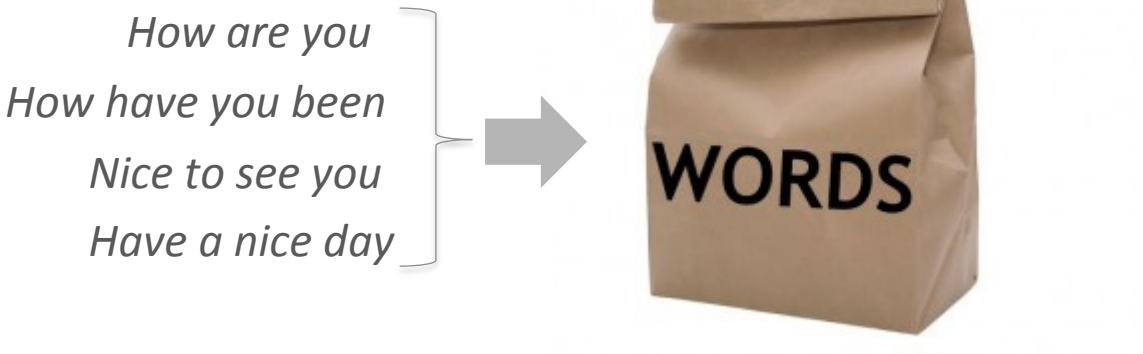
Each representation has only a single ‘1’ with all remaining 0s.

COUNT based WORD REPRESENTATION

Bag of Words (BOW)

- A bag-of-words model (BoW) is a representation of text that describes **the occurrence of words** within a document. It involves two things:
  - A vocabulary of known words.
  - A measure of the presence of known words.

- It is called a "**bag**" of words, because any information about the **order or structure of words in the document is discarded.** The model is only concerned with whether known words occur in the document.

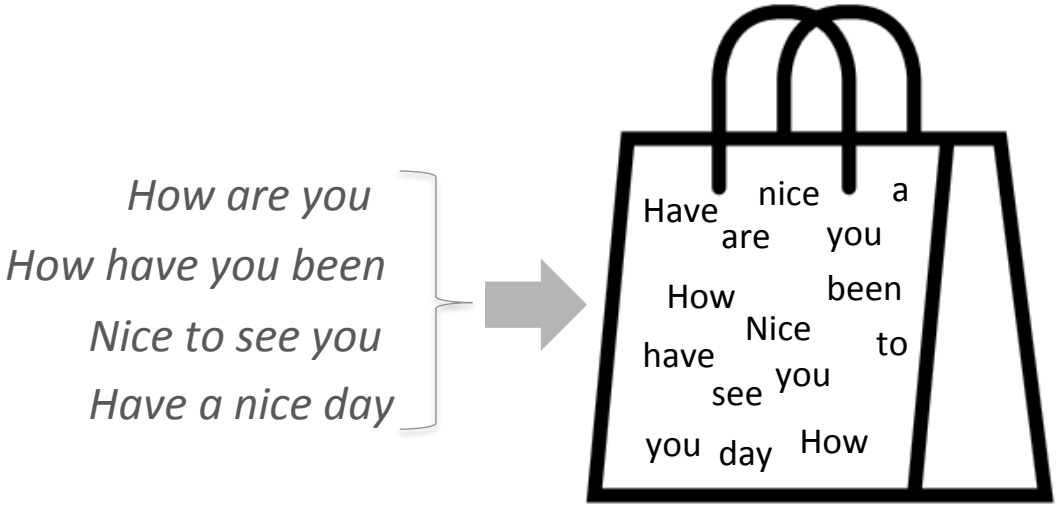


COUNT based WORD REPRESENTATION

Bag of Words (BOW)

- A bag-of-words model (BoW) is a representation of text that describes **the occurrence of words** within a document. It involves two things:
  - A vocabulary of known words.
  - A measure of the presence of known words.

- It is called a “bag” of words, because any information about the **order or structure of words in the document is discarded**



3 COUNT based WORD REPRESENTATION

Bag of Words (BOW)

A vocabulary of known words

a	are	been	day	have	how	nice	see	to	you

* **WO** = occurrence of words

[WO*a*, WO*are*, WO*been*, WO*day*, WO*have*, WO*how*, WO*nice*, WO*see*, WO*to*, WO*you*]

How are you = [0, 1, 0, 0, 0, 1, 0, 0, 0, 1]

How have you been = [0, 0, 1, 0, 1, 1, 0, 0, 0, 1]

Nice to see you = [0, 0, 0, 0, 0, 0, 1, 1, 1, 1]

Have a nice day = [1, 0, 0, 1, 1, 0, 1, 0, 0, 0]

Total Frequency = [1, 1, 1, 1, 2, 2, 2, 1, 1, 3]

a	are	been	day	have	how	nice	see	to	you
1	1	1	1	2	2	2	1	1	3

COUNT based WORD REPRESENTATION

Why use BoW?

- The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document.

Problem with BoW

- Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged ("this is interesting" vs "is this interesting").

*S1= I **love** you but you **hate** me*

*S2= I **hate** you but you **love** me*

COUNT based WORD REPRESENTATION

Term Frequency-Inverse Document Frequency

- Term Frequency-Inverse Document Frequency (TF-IDF) is a way of representing ***how important a word*** *is to a document in a collection or corpus*.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$w_{i,j}$ = *weight of term i in document j*

$tf_{i,j}$ = *number of occurances of term i in document j*

N = *total number of documents*

df_i = *number of documents containing term i*

- The ***Term Frequency*** is a count of how many times a word occurs in a given document (synonymous with bag of words)
- The ***Inverse Document Frequency*** is the number of times a word occurs in a corpus of documents

3 COUNT based WORD REPRESENTATION

Term Frequency

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Like BoW

$tf_{i,j}$ = *number of occurances of term i in document j*

Document #1: I like apple

Document #2: I like banana

Document #3: Sweet and yellow banana banana

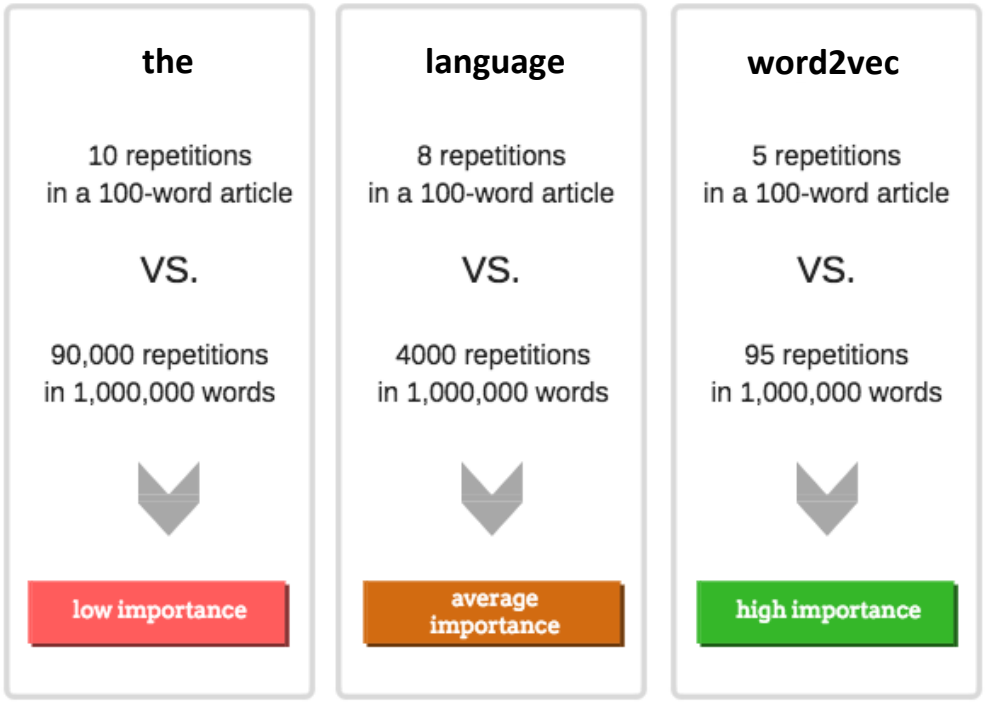
Document #4: Sweet fruit

	and	apple	banana	fruit	I	like	sweet	yellow
D#1	0	1	0	0	1	1	0	0
D#2	0	0	1	0	1	1	0	0
D#3	1	0	2	0	0	0	1	1
D#4	0	0	0	1	0	0	1	0

COUNT based WORD REPRESENTATION

Can we use Term Frequency Only?

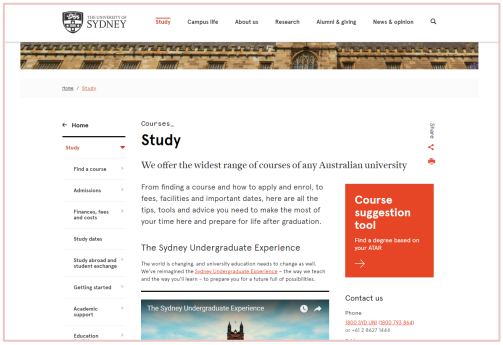
- It is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones.



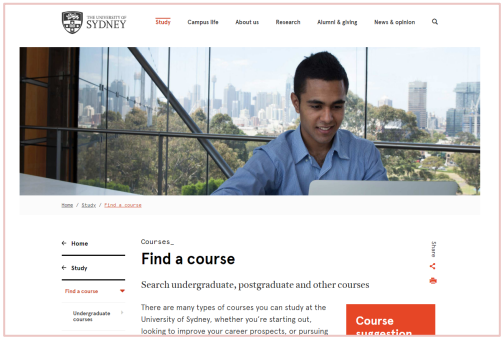
COUNT based WORD REPRESENTATION

Can we use Term Frequency Only?

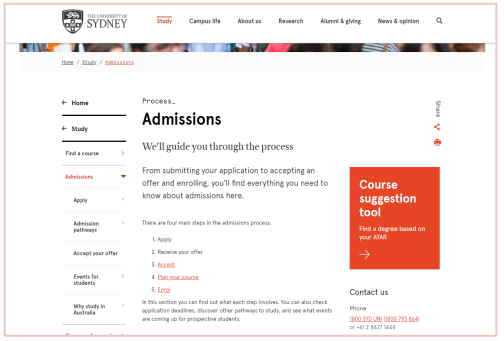
University of Sydney Website



Webpage#1



Webpage#2



Webpage#3

COUNT based WORD REPRESENTATION

Inverse Document Frequency

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

N = total number of documents

df_i = number of documents containing term i

Document #1: I like apple

Document #2: I like banana

Document #3: Sweet and yellow banana banana

Document #4: Sweet fruit

N = 4

	and	apple	banana	fruit	I	like	sweet	yellow
df	1	1	2	1	2	2	2	1

COUNT based WORD REPRESENTATION

Inverse Document Frequency

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$1+df_i$ *sometimes, why?*

N = total number of documents

df_i = number of documents containing term i

Document #1: I like apple

Document #2: I like banana

Document #3: Sweet and yellow banana banana

Document #4: Sweet fruit

N = 4

	and	apple	banana	fruit	I	like	sweet	yellow
df	1	1	2	1	2	2	2	1
idf (with 1+*dfi*)	Inv(4/(1+1)) =0.693147	Inv(4/(1+1)) =0.693147	Inv(4/(2+1)) =0.287682	Inv(4/(1+1)) =0.693147	Inv(4/(2+1)) =0.287682	Inv(4/(2+1)) =0.287682	Inv(4/(2+1)) =0.287682	Inv(4/(1+1)) =0.693147

We use a natural logarithm to the base of the mathematical constant e.
where e is an irrational and transcendental number approximately equal to 2.718281828459

3 COUNT based WORD REPRESENTATION

Term Frequency Inverse Document Frequency

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$ ← $1+df_i$

$w_{i,j}$ = *weight of term i in document j*

Document #1: I like apple

Document #2: I like banana

Document #3: Sweet and yellow banana banana

Document #4: Sweet fruit

	and	apple	banana	fruit	I	like	sweet	yellow
D#1	0	0.693147	0	0	0.287682	0.287682	0	0
D#2	0	0	0.287682	0	0.287682	0.287682	0	0
D#3	0.693147	0	0.575364	0	0	0	0.287682	0.693147
D#4	0	0	0	0.693147	0	0	0.287682	0

0 LECTURE PLAN

THE UNIVERSITY OF SYDNEY

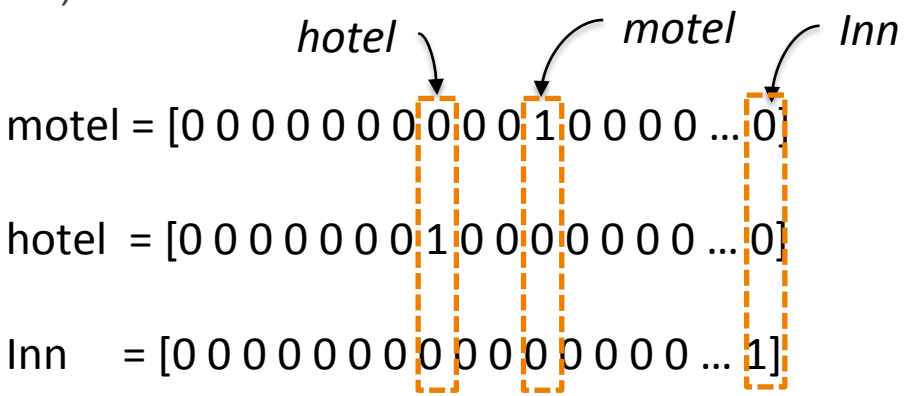
Lecture 2:

1. Lab Info
2. Word Meaning and Representation
3. Count based Word representation
    1. One-hot encoding
    2. Bag of Words
    3. Term Frequency-Inverse Document Frequency
4. **Prediction based Word representation**
    1. Word2Vec
    2. FastText
    3. Glove Word Vector
5. Next Week Preview

COUNT based WORD REPRESENTATION

Sparse Representation

With COUNT based word representation (especially, one-hot vector), linguistic information was represented with sparse representations (high-dimensional features)

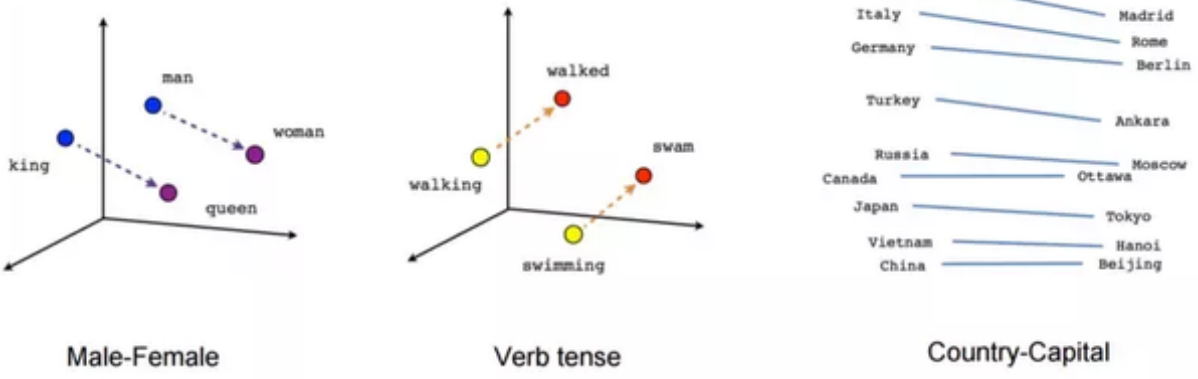


However, with the recent popularity and success of **word embeddings (low dimensional, distributed representations)**, **neural-based models have achieved superior results on various language-related tasks** as compared to traditional machine learning models with high-dimensional features.

Prediction based Word representation

Word2Vec : Represent the Word Similarity!

- How to represent the word similarity with dense vector



Male-Female | Verb tense | Country-Capital

- Try this with word2vec

Word Algebra

Enter all three words, the first two, or the last two and see the words that result.

tokyo + (australia - sydney) = Get result

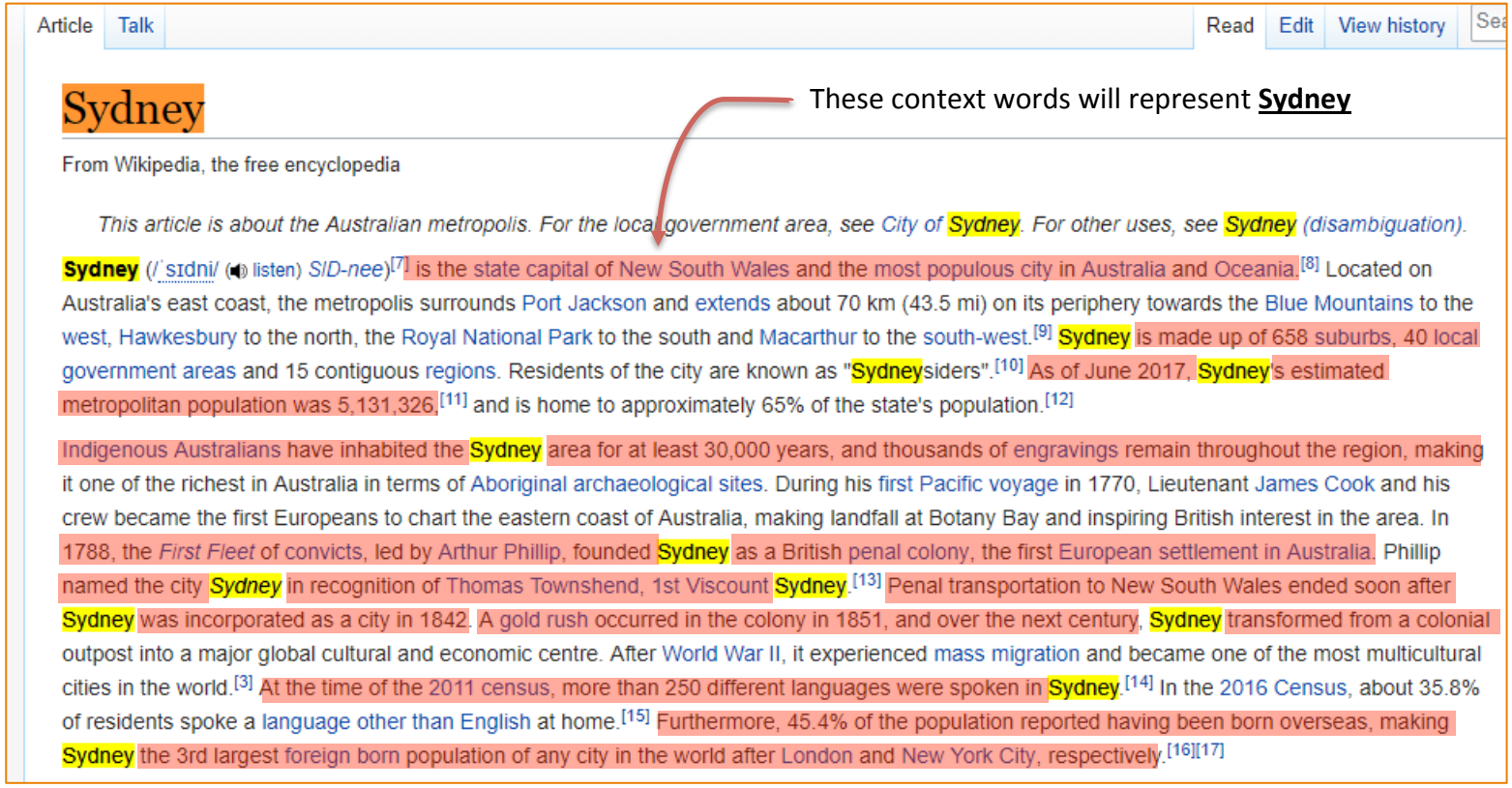
japan 0.8285006143050972

Reference: http://turbomaze.github.io/word2vecjson/
http://bionlp-www.utu.fi/wv_demo/

Prediction based Word representation

Word Representations in the context

- When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window)
- Use the many contexts of w to build up a representation of w



Prediction based Word representation

Distributed Representation

It substitute the word into a fixed-length numeric vector so that words of similar meanings have similar numeric vectors.

One hot vector

Sydney = [0, 0, 0, 0, 1, 0, 0, 0, … , 0]

* If 10,000 words are in the vocab, only one 1, the rest 9,999 0s

Distributed Representation

It is worth re-highlighting the training concept that in training the **numeric vector of a word (let's call it the center word),** the vector of the **center word** is optimized to *predict the surrounding context words*

Sydney = [0.2 0.3 0.5 0.7 0.2, …,0.2]

* It does not include all 10,000 words (just a fixed length numeric vector)

Prediction based Word representation

Word2Vec

Word2vec can utilize either of two model architectures to produce a distributed representation of words:

1. **Continuous Bag of Words (CBOW)**
   Predict center word from (bag of) context words

2. **Continuous Skip-gram**
   Predict context (“outside”) words (position independent) given center word

Prediction based Word representation

Word2Vec with Continuous Bag of Words (CBOW)

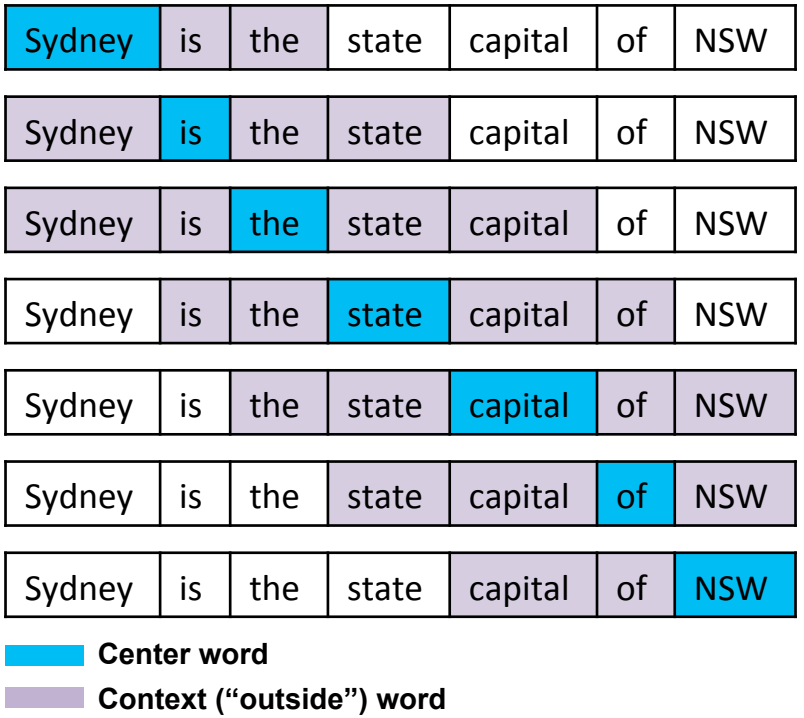
Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}

Assumption

- Predict the center word 'state'
- Window size (2)
  - Window sliding



Prediction based Word representation

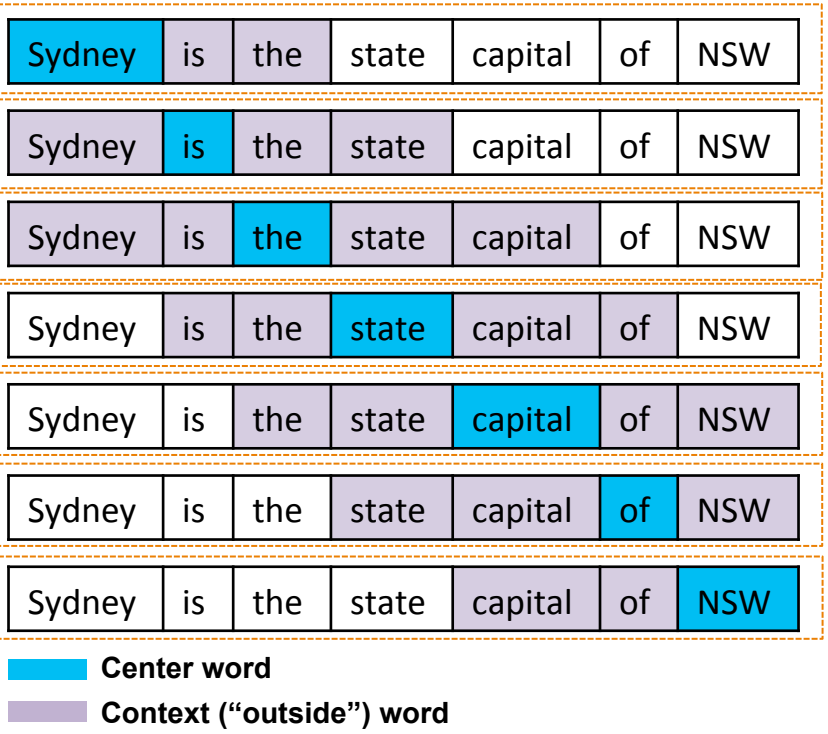
Word2Vec with Continuous Bag of Words (CBOW)

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

Using window slicing, develop the training data

Center word	Context ("outside") word
[1,0,0,0,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0]
[0,1,0,0,0,0,0]	[1,0,0,0,0,0,0], [0,0,1,0,0,0,0], [0,0,0,1,0,0,0]
[0,0,1,0,0,0,0]	[1,0,0,0,0,0,0], [0,1,0,0,0,0,0] [0,0,0,1,0,0,0], [0,0,0,0,1,0,0]
[0,0,0,1,0,0,0]	[0,1,0,0,0,0,0], [0,0,1,0,0,0,0] [0,0,0,0,1,0,0], [0,0,0,0,0,1,0]
[0,0,0,0,1,0,0]	[0,0,1,0,0,0,0], [0,0,0,1,0,0,0] [0,0,0,0,0,1,0], [0,0,0,0,0,0,1]
[0,0,0,0,0,1,0]	[0,0,0,1,0,0,0], [0,0,0,0,1,0,0] [0,0,0,0,0,0,1]
[0,0,0,0,0,0,1]	[0,0,0,0,1,0,0], [0,0,0,0,0,1,0]



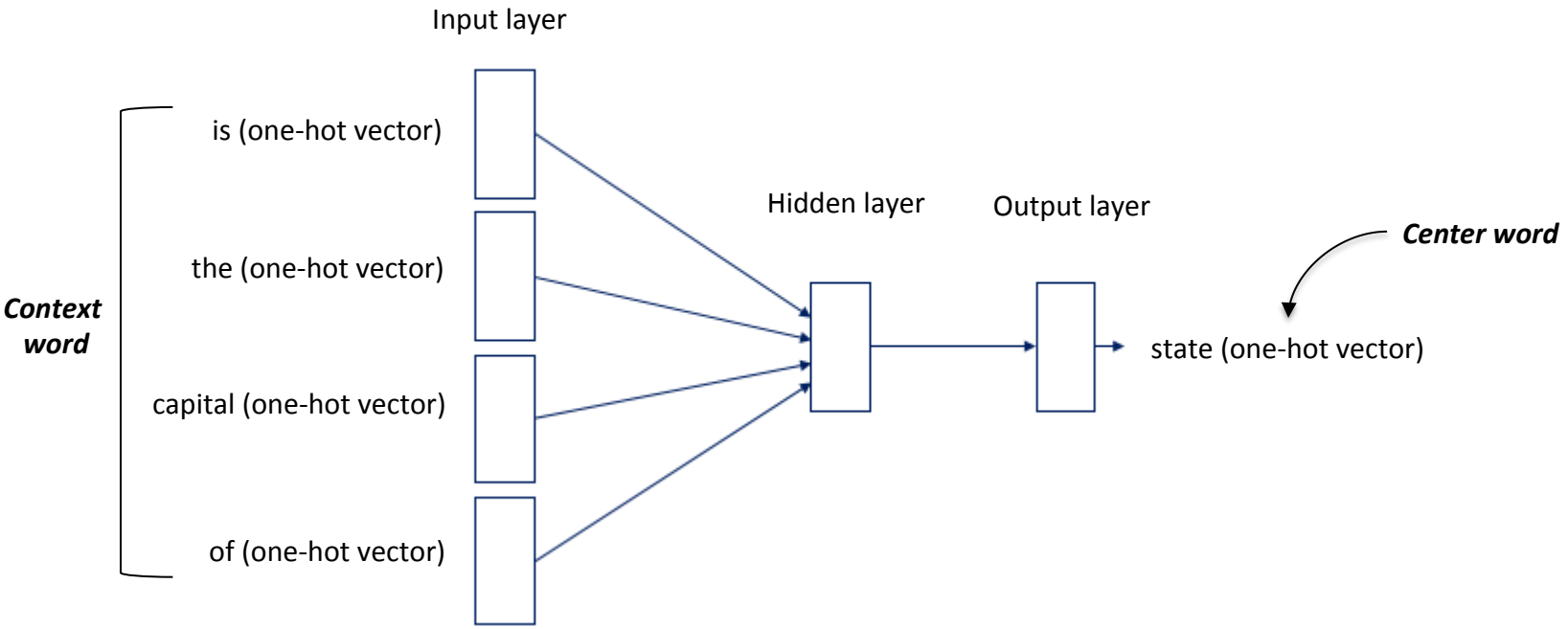
Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}



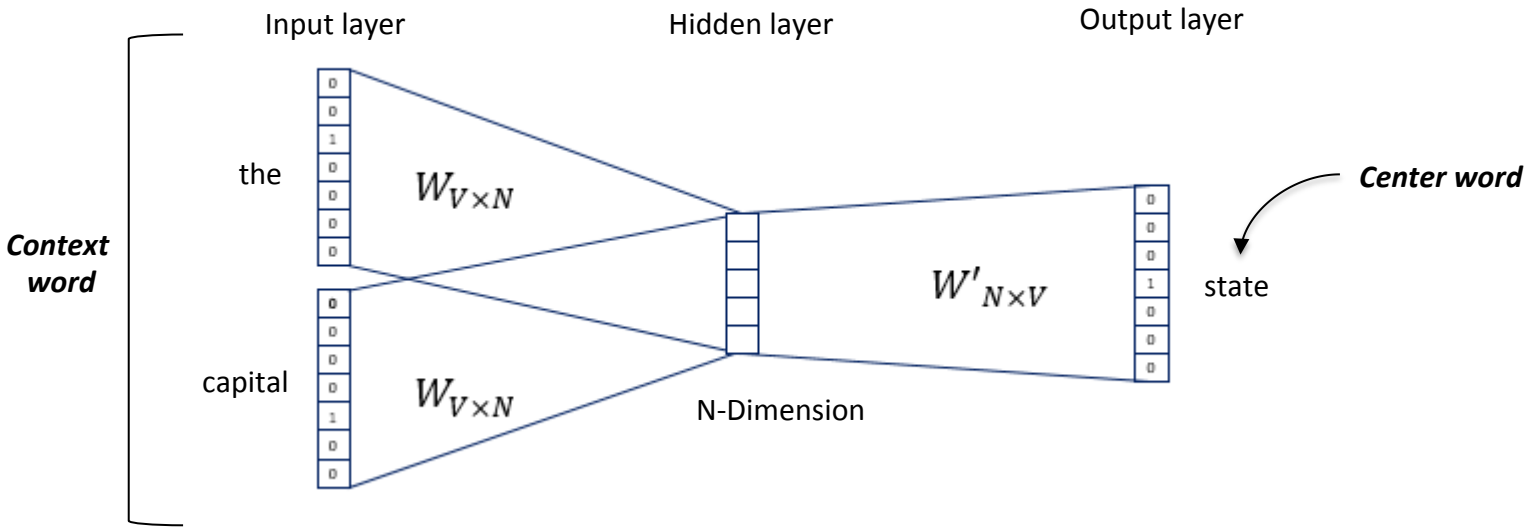
Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}



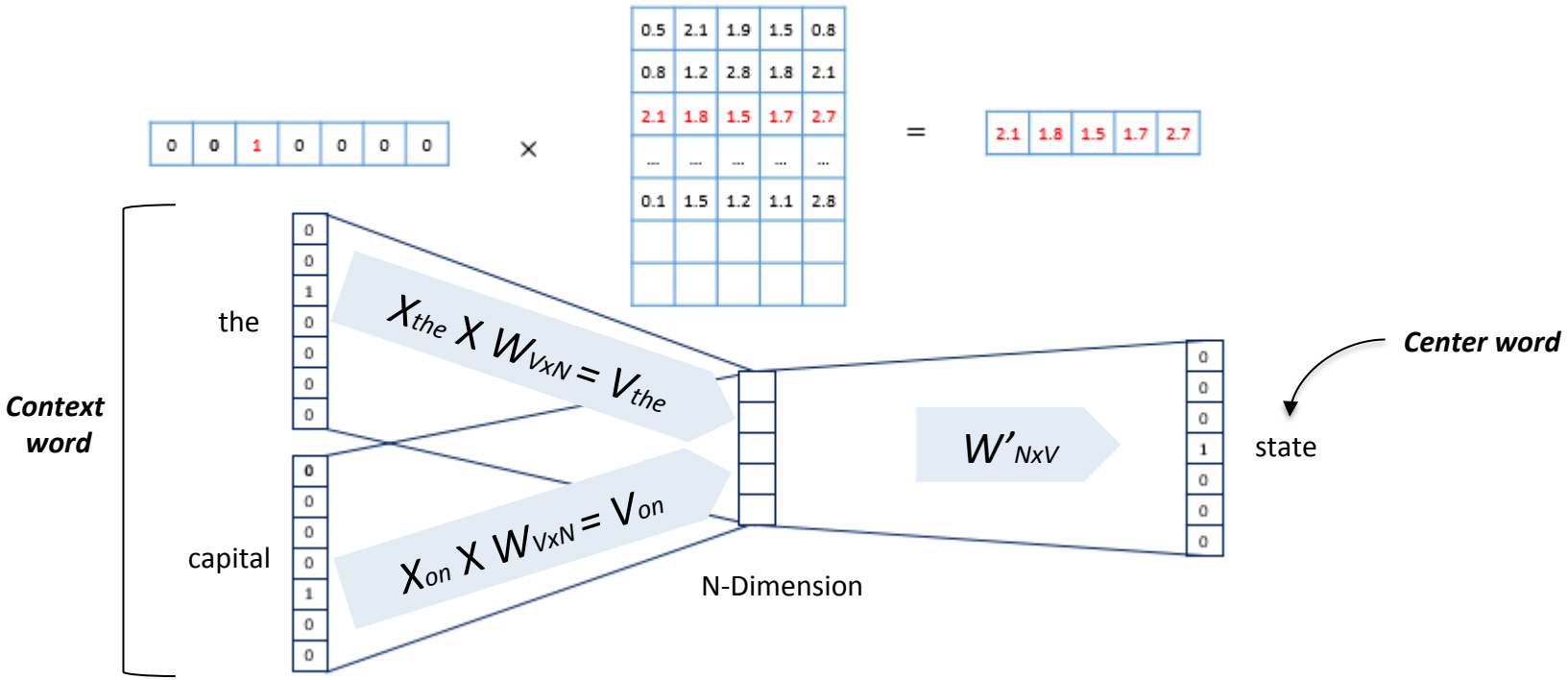
Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}



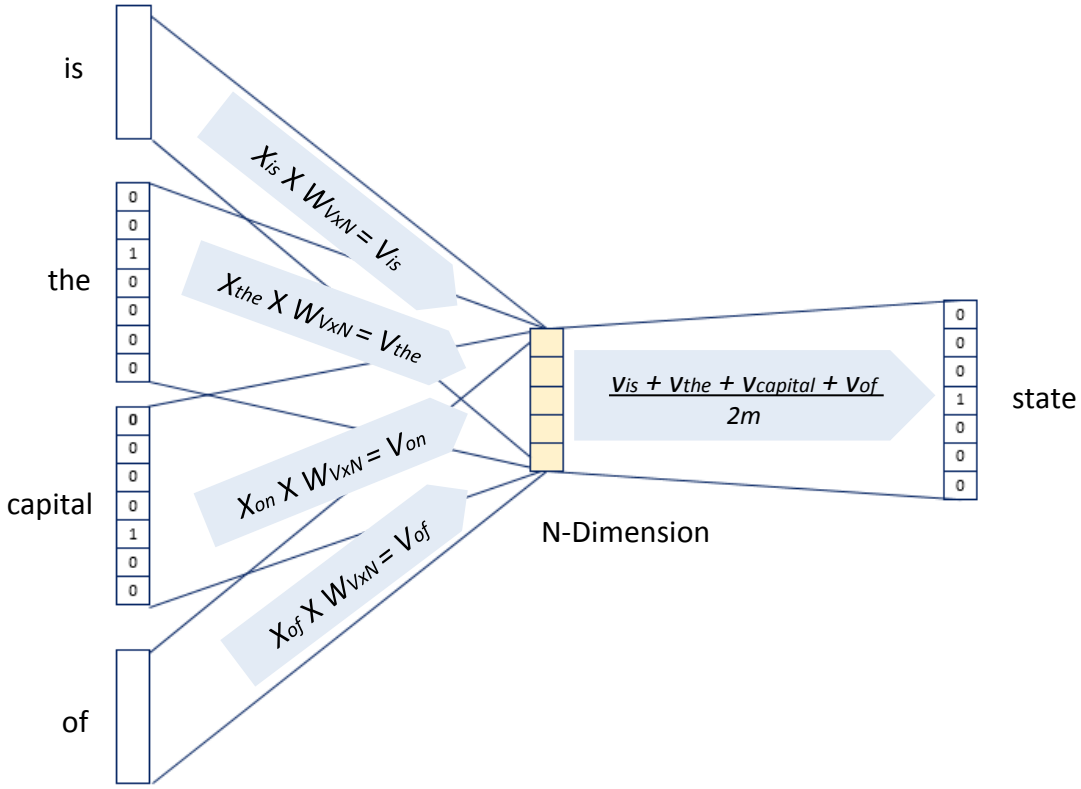
Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}



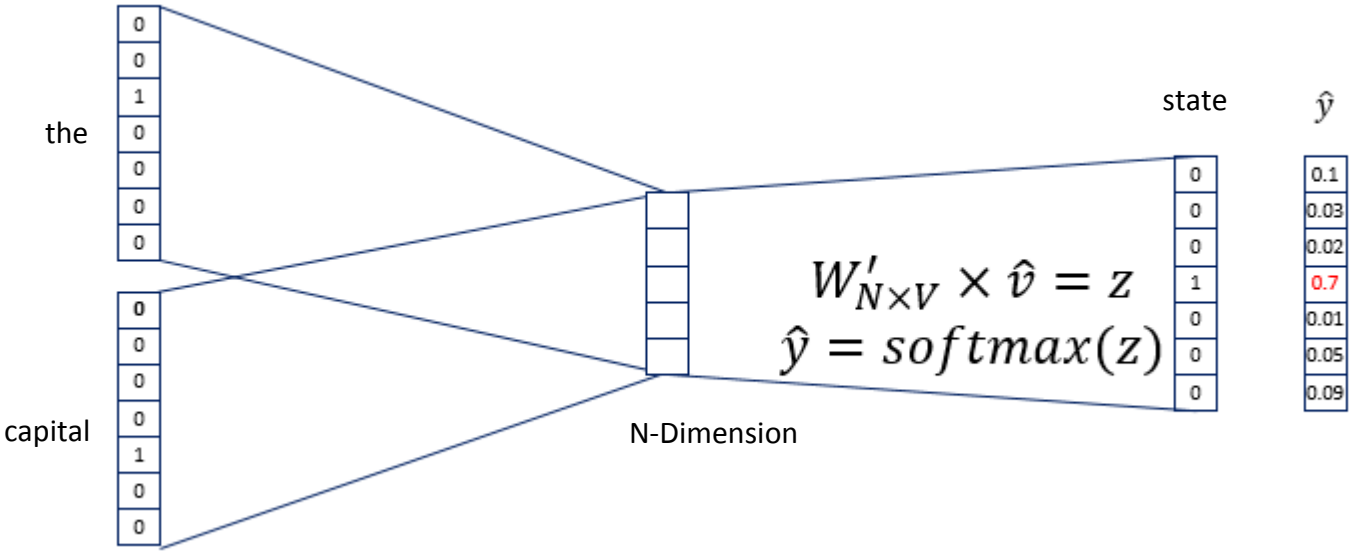
Prediction based Word representation

CBOW – Neural Network Architecture

Predict center word from (bag of) context words

Sentence: "Sydney is the state capital of NSW"

{'Sydney', 'is', 'the', 'capital', 'of', 'NSW'}

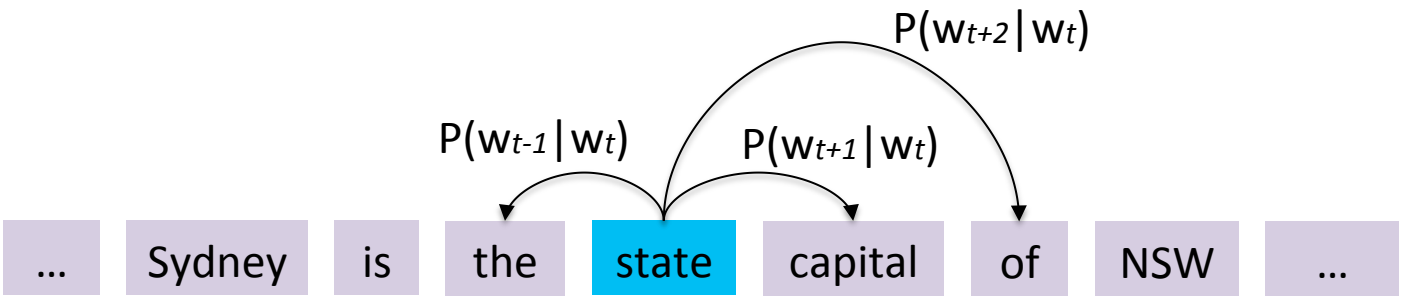


Prediction based Word representation

Skip Gram

Predict context ("outside") words (position independent) given center word

Sentence: "Sydney is the state capital of NSW"

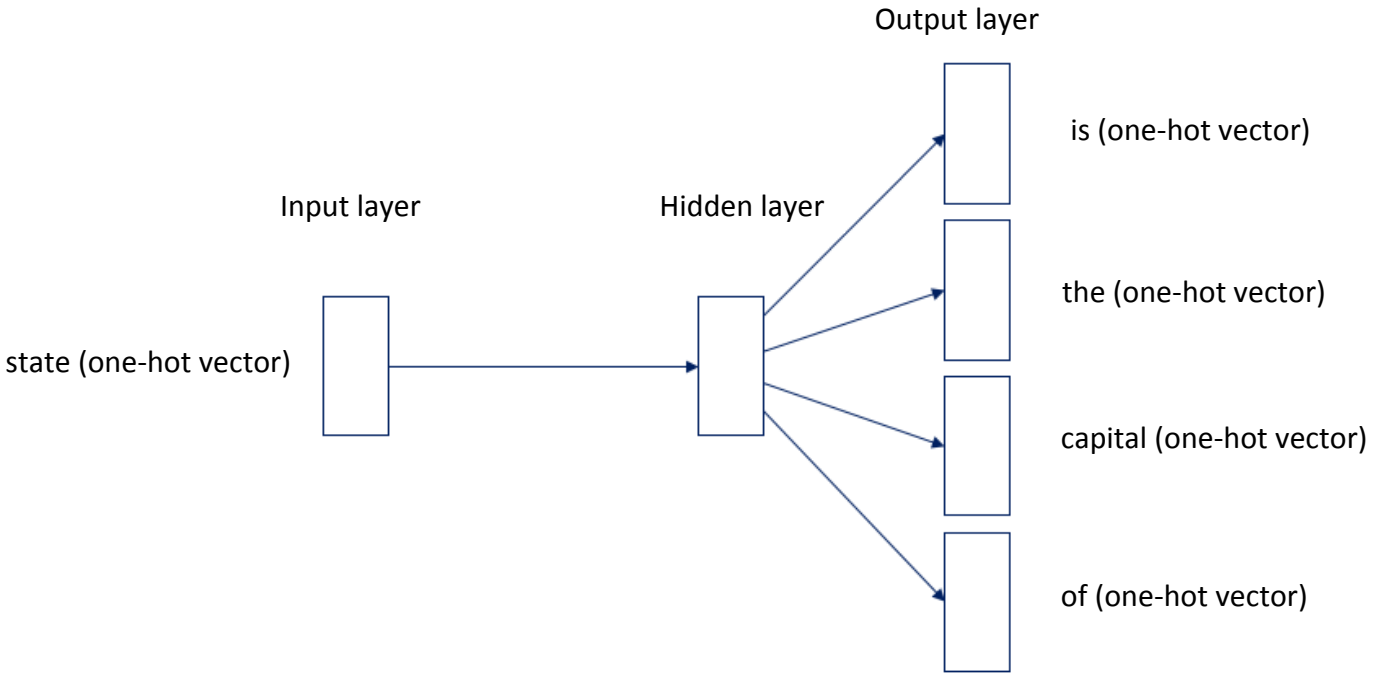


Prediction based Word representation

Skip Gram

Predict context ("outside") words (position independent) given center word

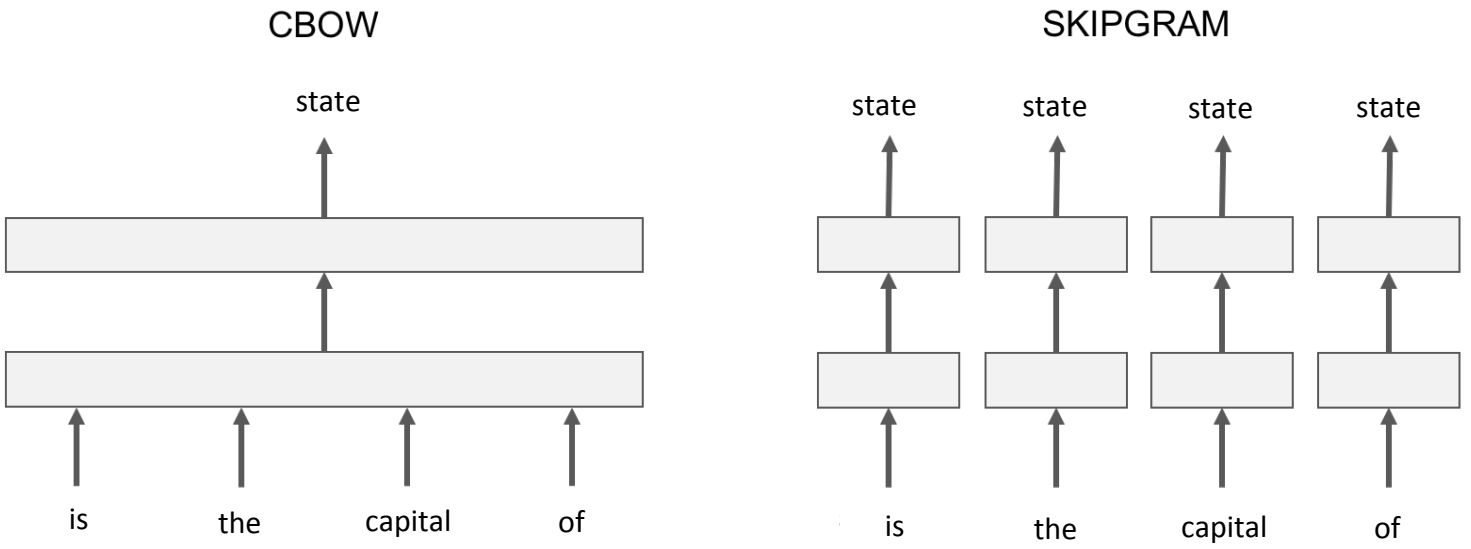
Sentence: "Sydney is the state capital of NSW"



Prediction based Word representation

CBOW vs Skip Gram Overview

Sentence: "Sydney is the state capital of NSW"



Prediction based Word representation

Word2Vec Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors

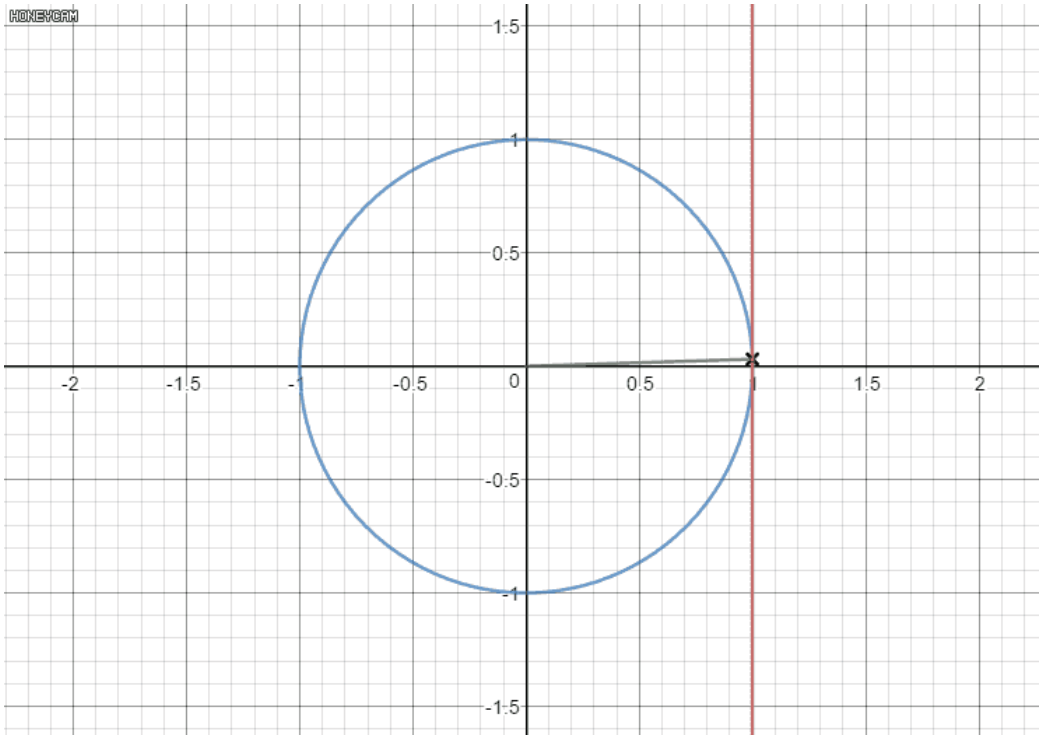
Idea:

- Have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position *t* in the text, which has a center word *c* and context ("outside") words *o*
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Prediction based Word representation

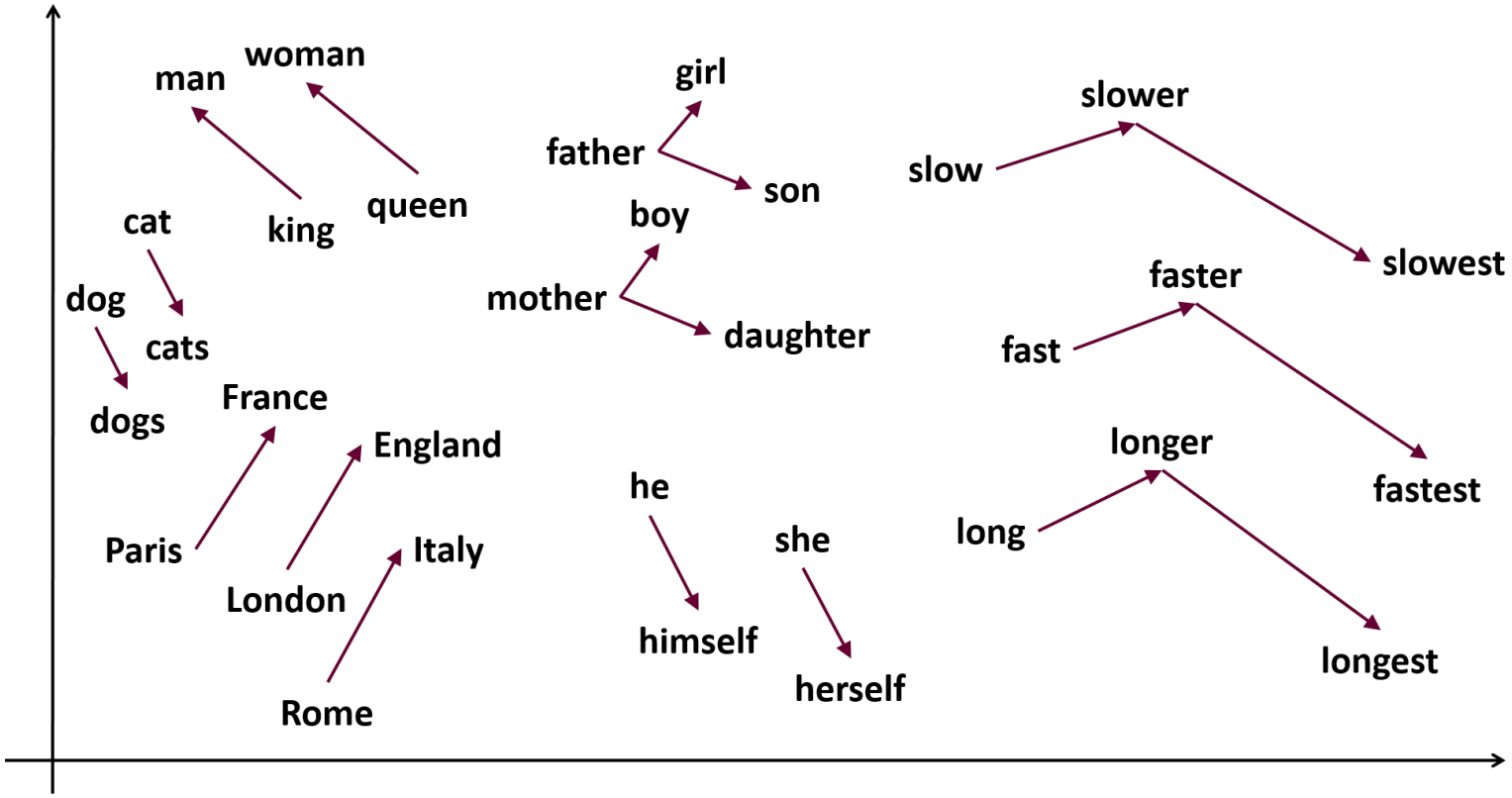
Word2Vec Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors



Prediction based Word representation

Let's try some Word2Vec!



Gensim: https://radimrehurek.com/gensim/models/word2vec.html

Resources: https://wit3.fbk.eu/
https://github.com/3Top/word2vec-api#where-to-get-a-pretrained-models

Prediction based Word representation

Limitation of Word2Vec

Issue#1: Cannot cover the morphological similarity

- Word2vec represents every word as an independent vector, even though many words are morphologically similar, like: teach, teacher, teaching

Issue#2: Hard to conduct embedding for rare words

- Word2vec is based on the Distribution hypothesis. Works well with the frequent words but does not embed the rare words.

  (same concept with the under-fitting in machine learning)

Issue#3: Cannot handle the Out-of-Vocabulary (OOV)

- Word2vec does not work at all if the word is not included in the Vocabulary

Prediction based Word representation

FastText

- Deal with this Word2Vec Limitation
- Another Way to transfer *WORDS* to *VECTORS*

- FastText is a library for learning of word embeddings and text classification created by Facebook's AI Research lab. The model allows to create an unsupervised learning or supervised learning algorithm for obtaining vector representations for words.

- Extension to Word2Vec
  - Instead of feeding individual words into the Neural Network, FastText breaks words into several n-grams (sub-words)

https://fasttext.cc/

Prediction based Word representation

FastText with N-gram Embeddings

- N-grams are simply all combinations of adjacent words or letters of length n that you can find in your source text. For example, given the word *apple*, all 2-grams (or "bigrams") are ***ap, pp, pl,*** and ***le***

- The tri-grams (n=3) for the word apple is ***app, ppl,*** and ***ple*** (ignoring the starting and ending of boundaries of words). The word embedding vector for apple will be the sum of all these n-grams.

- After training the Neural Network (either with skip-gram or CBOW), we will have word embeddings for all the n-grams given the training dataset.

- Rare words can now be properly represented since it is highly likely that some of their n-grams also appears in other words.

https://fasttext.cc/

Prediction based Word representation

Word2Vec VS FastText : electrofishing

Find synonym with Word2vec

```
from gensim.models import Word2Vec
cbow_model = Word2Vec(sentences=result, size=100, window=5, min_count=5, workers=4, sg=0)

a=cbow_model.wv.most_similar("electrofishing")
pprint.pprint(a)
```

Find synonym with FastText

```
from gensim.models import FastText
FT_model = FastText(result, size=100, window=5, min_count=5, workers=4, sg=0)

a=FT_model.wv.most_similar("electrofishing")
pprint.pprint(a)
```

https://fasttext.cc/

Prediction based Word representation

Global Vectors (GloVe)

- Deal with this Word2Vec Limitation

"Methods like skip-gram may do better on the analogy ask, but they poorly utilize the statistics of the corpus since they train on separate local context windows instead of on global co-occurrence counts."

(PeddingLon et al., 2014)

- Focus on the Co-occurrence

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

https://nlp.stanford.edu/projects/glove/

Prediction based Word representation

Limitation of Prediction based Word Representation

- I like _____

  apple *banana* *fruit*

- Training dataset reflect the word representation result
  - The word similarity of the word ‘software’ the model learned by Google News corpus can be different from the one from Twitter.

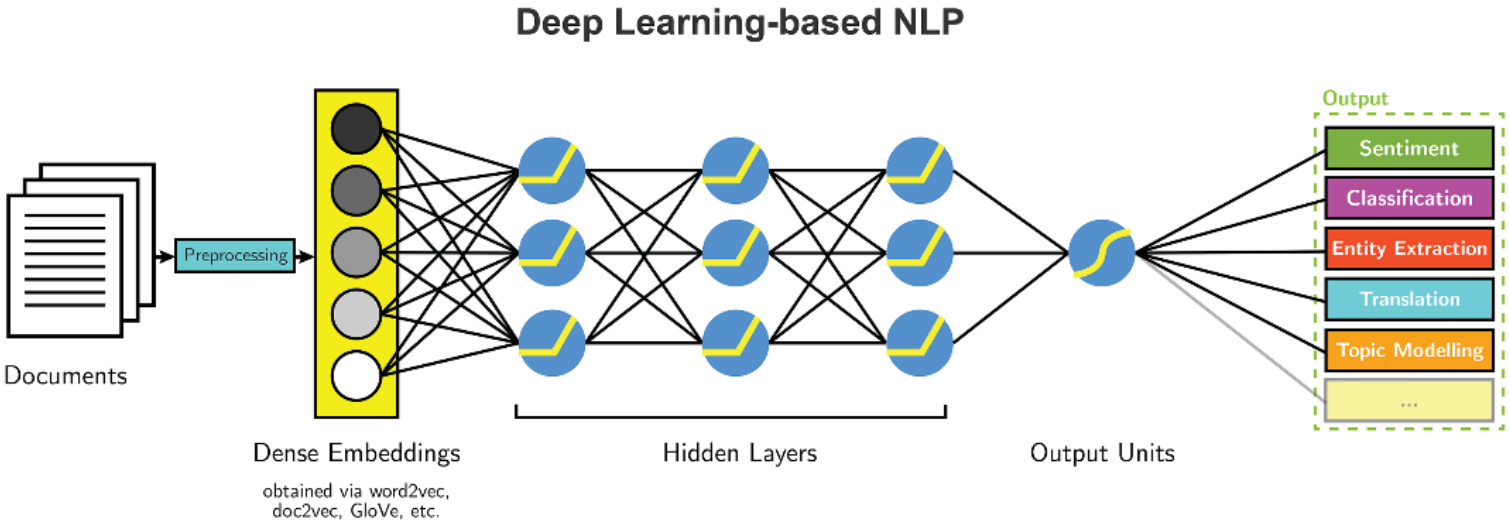
https://nlp.stanford.edu/projects/glove/

NEXT WEEK PREVIEW…

Word Embeddings

- How to Make Training Dataset for Natural Language Processing

Machine Learning for Natural Language Processing



Reference

Reference for this lecture

- Deng, L., & Liu, Y. (Eds.). (2018). Deep Learning in Natural Language Processing. Springer.
- Rao, D., & McMahan, B. (2019). Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. " O'Reilly Media, Inc.".
- Manning, C. D., Manning, C. D., & Schütze, H. (1999). Foundations of statistical natural language processing. MIT press.
- Manning, C 2017, Introduction and Word Vectors, Natural Language Processing with Deep Learning, lecture notes, Stanford University

Word2vec

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

FastText

- Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.
- Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in pre-training distributed word representations. arXiv preprint arXiv:1712.09405.